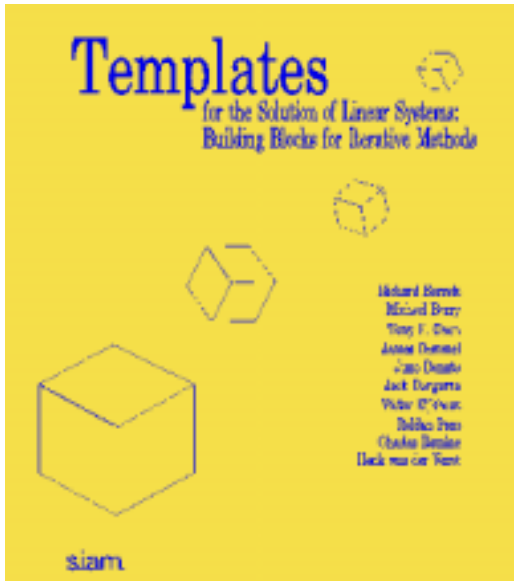
Templates for the Solution of Large and Sparse Linear Systems and Eigenvalue Problems



Royalties from the sale of these books are contributed to the SIAM Student Travel Fund.

Participants

- Linear systems:

  R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst.

- Eigenproblems:

  Z. Bai, T.-Y. Chen, D. Day, J. Demmel, J. Dongarra, A. Edelman, T. Ericsson, R. Freund, M. Gu, B. Kågström, A. Knyazev, P. Koev, T. Kowalski, R. Lehoucq, R.-C. Li, X. Li, R. Lippert, K. Maschhoff, K. Meerbergen, R. Morgan, A. Ruhe, Y. Saad, G. Sleijpen, D. Sorensen, H. van der Vorst.

Participants

Why templates?

- Linear systems and eigenproblems come in many formulations, with many possible solutions
- Accuracy, speed, reliability, memory, all tradeoffs
- How do we distill our knowledge to advise a user on which algorithm to use?
- A template is a high-level description of an algorithm, suitable for understanding how it works, what parameters the user can tune to control efficiency and accuracy, and how to interpret the output, and has pointers to complete or partial implementations, perhaps in several languages or for several computer architectures.
- "Decision tree" to guide the correct choice of algorithms.

Intended readership

- Students and teachers
  1. simple but generally effective algorithms
  2. access to the latest developments in a form that is easy to explain and to understand.

  $\Rightarrow$ high level algorithm descriptions to acquire understanding of methods.
- General engineers and scientists
  1. easy-to-use,
  2. reliable software,
  3. reasonably efficient.

  $\Rightarrow$ decision tree and pointers to software adequate for their problems.
- Engineers and scientists in the high performance computing community
  1. high speed,
  2. access to algorithmic details for performance tuning,
  3. reliability for their problem.

  $\Rightarrow$ advice on performance tuning and assessing the accuracy of the output

What is a template?

A template will include some or all of the following information:

- A high-level description of an algorithm.
- A description of when the algorithm is effective, including conditions on the input, and estimates of the time, space or other resources required. If there are natural competitors, they will be referenced.
- A description of available refinements and user tunable parameters, as well as advice on when to use them.
- A way to assess the accuracy.
- Numerical examples, illustrating both easy and difficult cases.
- Pointers to complete or partial implementations, perhaps in several languages or for several computer architectures.
- Pointers to texts or journal articles for further information.

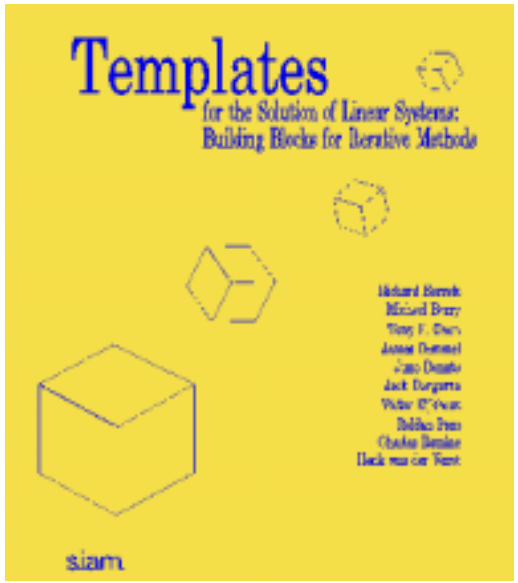
Templates for linear systems and eigenproblems

Linear systems (1994):

LSTHOME=http://www.netlib.org/linalg/html_templates/Templates.html

Eigenproblems (2000):

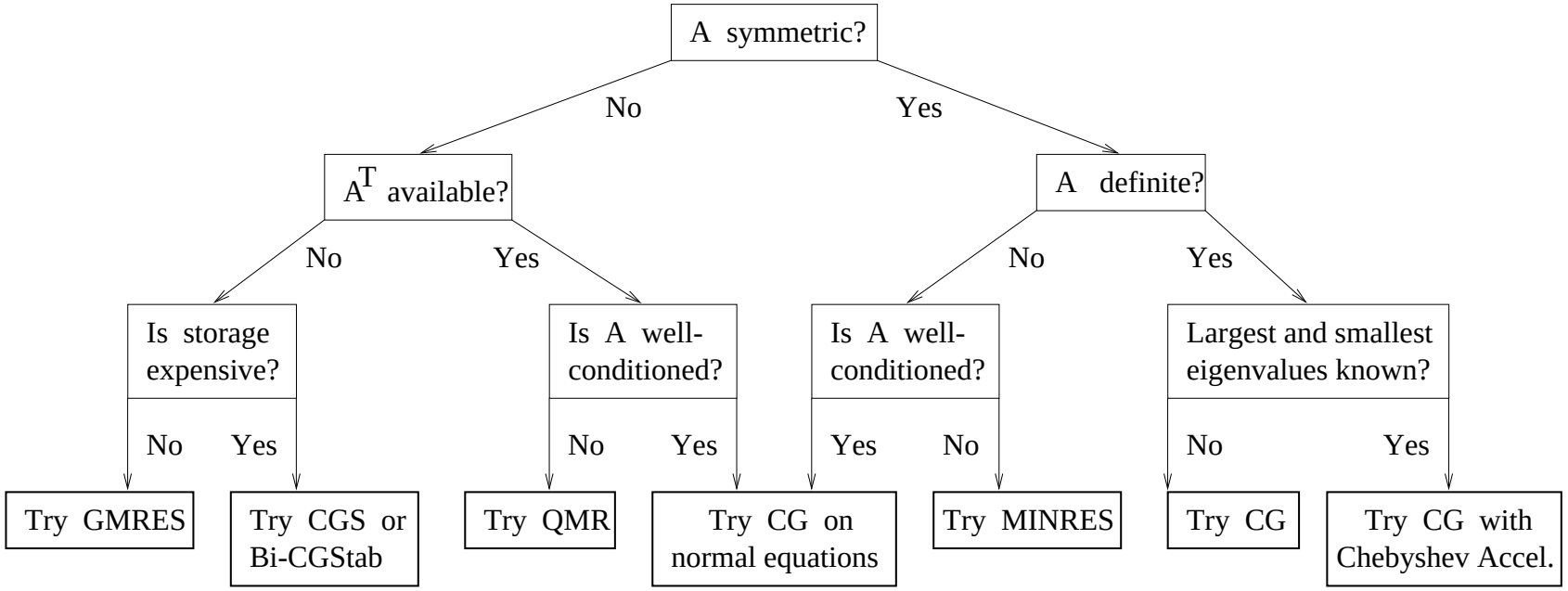
ETHOME = http://www.netlib.org/etemplates



Contents of Templates for linear systems

- Algorithms for
  - Jacobi, Gauss-Seidel, SOR, SSOR, ...
  - PCG, Bi-CG, GMRES, QMR, ...
  - Preconditioners (Jacobi, SSOR, Incomplete factorizations, ...)
- Convergence test
- Hints for parallelism
- Advice on choice of algorithm (next slide)
- Common issues: sparse data structure, ...
- Unified codes, Matlab, Fortran, C, C++ (see Software Respository)
- Available in Matlab, Aztec, PETSc, ...

"Decision tree" for linear systems



"Decision tree" for eigenproblems

Level 1 *Mathematical properties of the problem, such as*

1. *Is it a Hermitian (real symmetric, self adjoint) or a non-Hermitian eigenproblem?*
2. *Is it a standard problem involving only one matrix, a generalized problem with two matrices?*

Level 2 Desired spectral information:

1. Do we need
   - the smallest eigenvalue,
   - few eigenvalues at either end of the spectrum,
   - a subset of eigenvalues "inside" the spectrum,
   - or all eigenvalues?
2. Do we want
   - associated eigenvectors,
   - invariant subspaces,
   - or other quantities?
3. How much accuracy is desired?

"Decision tree" for eigenproblems, cont.

Level 3 *Available Operations and their costs:*

1. *Can we store the full matrix as an array and perform a similarity transformation on it?*
2. *Can we solve a linear system with the matrix (or shifted matrix) by a direct factorization routine, or perhaps an iterative method?*
3. *Or can we only multiply a vector by the matrix, and perhaps by its transpose?*
4. *If several of these operations are possible, what are their relative costs?*

Contents of Templates for Eigenproblems

Chapter 2: A Brief Tour of Eigenproblems

- Categorizing eigenproblems:
  1. Hermitian eigenproblems: $Ax = \lambda x$, $A^* = A$
  2. Generalized Hermitian eigenproblems: $Ax = \lambda Bx$, $A^* = A$, $B^* = B > 0$
  3. Singular value decomposition: $A = U\Sigma V^*$
  4. Non-Hermitian eigenproblems: $Ax = \lambda x$, $A^* \neq A$
  5. Generalized non-Hermitian eigenproblems: $Ax = \lambda Bx$,
  6. Nonlinear eigenproblems: e.g.,
     $(\lambda^2 M + \lambda D + K)x = 0$,
     $P(\lambda)x$

Chapter 2: A Brief Tour of Eigenproblems, cont.

- Essentials
  1. Definitions of eigenvalues and eigenvectors
  2. Definitions of eigenspaces
  3. Equivalences
  4. Eigendecompositions
  5. Conditioning
  6. Different ways of specifying an eigenproblem
  7. Related eigenproblems
  8. Model problems from vibrational analysis

Chapter 2: A Brief Tour of Eigenproblems, cont.

Converting among Eigenproblems:

- Under Hermitian eigenproblems $Ax = \lambda x$:

  If $A = B^*B$, try SVD of B

- Under SVD of general A:

  If $A = BC^{-1}$ with C ill-conditioned, try GSVD of B, C.

- Under regular generalized eigenproblems $Ax = \lambda Bx$:

  If $A^* = A$ and $B^* = B$, $\alpha A + \beta B$ pos. def., try eigenproblem of A and $\alpha A + \beta B$.

- Under non-Hermitian eigenproblems $Ax = \lambda x$:

  If $B = \alpha SAS^{-1}$ for known simple α and S, try Hermitian eigenproblems of B

- Under polynomial eigenproblems, try (block) companion matrix.

Chapter 4: Hermitian Eigenvalue Porblems (An Example)

Section 4.1 Introduction (of Hermitian Eigenporblems)

- Review of basic definitions and concepts
- List of available methods and main features
- Summary of choices

Algorithm	Variant		Eigenvalues Sought			Storage	
	Appl	Orth	IE	CE	M	# vec	Fact
Power	Dir		Yes	Very slow	No	2	-
	SI		-	Yes	Yes	2	LU
Subspace iter	Dir	FO	Yes	Slow	No	Moderate	-
	SI	FO	-	Yes	Yes	Moderate	LU
Lanczos	Dir	local	Yes	No	No	3	-
	Dir	SO	Yes	Slow	No	Many	-
	SI	FO	-	Yes	Yes	Moderate	LU
IR Lanczos	Dir	FO	Yes	Slow	No	Few	-
	SI	FO	-	Yes	Yes	Fewer	LU
Band Lanczos	Dir	FO	Yes	Yes	No	Many	-
	SI	FO	-	Yes	Yes	Moderate	LU
Jac-Dav	Dir	FO	Slow	Slow	No	Few	-
	Prec	FO	Yes	Yes	Slow	Few	ILU
	SI	FO	-	Yes	Yes	Few	LU

Section 4.1 Introduction (of Hermitian Eigenporblems), cont.

▷ Matrix Preparation:

Dir = direct application: $v = Au$

SI = Shift-and-invert, $v = (A - \sigma)^{-1}u$,

Prec = application with a preconditioner

▷ Orthogonalization:

FO = full orthogonalization

SO = selected orthogonalization

Local = local orthogonalization

▷ Eigenvalue Sought:

IE = one or a few isolated eigenvalues at the end of spectrum

CE = one or several clustered eigenvalues at the end of spectrum

M = some eigenvalues in the middle of spectrum

▷ Storage:

#vec = the number of vectors needed to store

Fact = extra matrix storage, for example, for sparse LU factorization

- Extra: e.g., counting the number of eigenvalues in $[\alpha, \beta]$

Section 4.4 Lanczos method

- Basic theory and formulation
- Algorithm templates

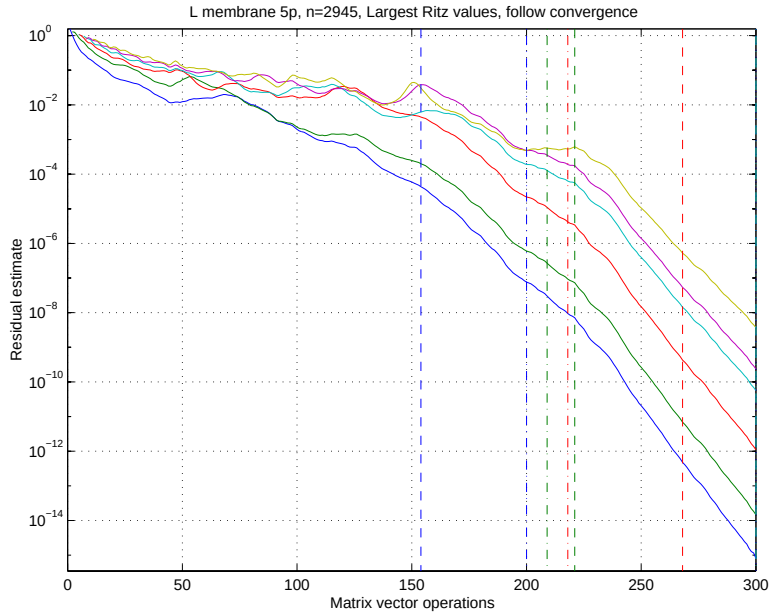
(1) start with $r = v$, starting vector
(2) $\beta_0 = \|r\|_2$
(3) **for** $j = 1, 2, \ldots,$ **until** Convergence,
(4) $v_j = r/\beta_{j-1}$
(5) operate $r = Av_j$
(6) $r = r - v_{j-1}\beta_{j-1}$
(7) $\alpha_j = v_j^* r$
(8) $r = r - v_j\alpha_j$
(9) reorthogonalize if necessary
(10) $\beta_j = \|r\|_2$
(11) compute approximate eigenvalues $T_j = S\Theta^{(j)}S^*$
(12) test bounds for Convergence
(13) **end for**
(14) compute approximate eigenvectors $X = V_j\, S$

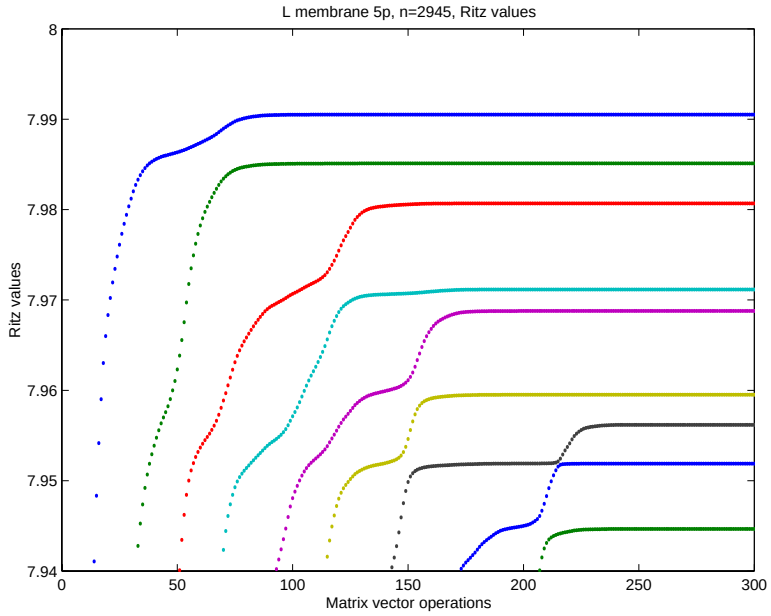
- Comments on some steps of algorithm
- Convergence properties
- Spectral transformation
- Reorthogonalization
- Software availability: LANCZOS, LASO, LANZ, ... $\Rightarrow$ ETHOME

Section 4.4 Lanczos method, cont.

Numerical illustration for an L-shaped membrane eigenproblem:

Convergence of residual norms and Ritz values:





Chapter 10 Common Issues

- Sparse matrix storage formats
  - Compressed row storage
  - Compressed column storage
  - Block compressed row storage
  - Compressed diagonal storage
  - Jagged diagonal storage
  - Skyline storage
- Matrix-vector and matrix multiplications
  - BLAS
  - Sparse BLAS
  - Fast matrix-vector multiplications for structured matrices

Chapter 10 Common Issues, cont.

- A brief survey of direct linear solvers
  - Direct solvers for dense matrices
  - Direct solvers for band matrices
  - Direct solvers for sparse matrices
  - Direct solvers for structured matrices
- A brief survey of iterative linear solvers
- Parallelism

ETHOME = http://www.netlib.org/etemplates

Templates for the Solution of Algebraic Eigenvalue Problems

A Practical Guide

Large-scale problems of engineering and scientific computing often require solutions of eigenvalue and related problems. This book gives a unified overview of theory, algorithms, and practical software for eigenvalue problems. The material is accessible for the first time to experts as well as many nonexpert users who need to choose the best state-of-the-art algorithms and software for their problems. Using an informal decision tree, just enough theory is introduced to identify the relevant mathematical structure that determines the best algorithm for each problem.

The algorithm "leaves" of the decision tree range from the classical QR algorithm, which is most suitable for small dense matrices, to iterative algorithms for very large generalized eigenvalue problems. Algorithms are presented in a unified style as templates, with different levels of detail suitable for readers ranging from beginning students to experts.

Royalties from the sale of this book are contributed to the SIAM Student Travel Fund.

- » Contributors
- » Online Book
- » Order the Book
- » How to Cite this Work
- » Software Repository
- » Of Things Not Treated
- » Feedback
- » Acknowledgments

Software repository for eigenproblems

≫ List by chapter

– Chapter 4 Hermitian Eigenvalue Problems

Section	Package name	Language	Comments
4.2	LAPACK, ScaLAPACK	F77, C, C++	direct methods
4.4	LANCZOS	F77	Lanczos and block Lanczos
4.4	LANZ	C, Fortran	Lanczos
4.4	LASO2	Fortran IV	block Lanczos
4.4	LANSO	F77	Lanczos
4.4	PLANSO	F77	parallel version of LANSO
4.4	PROPACK	F77, MATLAB	Lanczos method
4.5	ARPACK	F77, C++	IRLM
4.5	TRLAN	F90	thick restart Lanczos
4.7	JDQR	MATLAB	Jacobi-Davidson

– Chapter 5 Generalized Hermitian Eigenvalue Problems

– Chapter 6 Singular Value Decomposition

– Chapter 7 Non-Hermitian Eigenvalue Problems

– Chapter 8 Generalized Non-Hermitian Eigenvalue Problems

– Chapter 9 Nonlinear Eigenvalue Problems

– Chapter 10 Common Issues

– Chapter 11 Preconditioning Techniques

Software repository for eigenproblems, cont.

≫ List by method name

- Direct Methods
- Single- and Multiple-Vector Iteration Methods
- Arnoldi Methods
- Lanczos Methods

  * Hermitian eigenproblems:

Package name	Language	Comments
LANCZOS	Fortran 77	Lanczos and block Lanczos
LANZ	C, Fortran	SI, PRO
LASO2	Fortran IV	blocked Lanczos
LANSO/PLANSO	Fortran 77	PRO/parallel
TRLAN	Fortran 90	thick restart Lanczos
ARPACK/PARPACK	Fortran 77, C++	IRLM

  * Non-Hermitian eigenproblems:

Package name	Language	Comments
QMRPACK	Fortran 77	incl. complex symmetric
ABLEPACK	MATLAB	SI, full and partial reorth.
BLZPACK	Fortran 77	blocked
SILM	MATLAB	symmetric indefinite

Software repository for eigenproblems, cont.

  - Jacobi-Davidson Methods
  - Preconditioning Techniques

- Proprietary software known to us
- Test Matrix Market
- *New Software Entry Form* (welcome your contributions!)